EXHIBIT B

NetApp

ZFS Best Practices with NetApp

Guidelines for Configuring the ZFS File System with Network Appliance™ Storage Arrays.

Ganesh Kamath, Network Appliance, Inc.

June 2007 | TR-3603

Executive Summary

This technical report contains best practices and guidelines for configuring the new ZFS file system with Network Appliance storage systems.

1. What Is ZFS?

The Solaris™ ZFS file system is a new file system that fundamentally changes the way file systems are administered, with new features and benefits. ZFS has been designed to be robust, scalable, and simple to administer and combines volume manager and file system capabilities. A few features are:

- **Transactional Semantics**

  ZFS is a transactional file system, which means that the file system state is always consistent on disk. Data is never overwritten, and any sequence of operations is either entirely committed or entirely ignored. This mechanism means that the file system can never be corrupted through accidental loss of power or a system crash, so no need for a fsck equivalent exists. While the most recently written pieces of data might be lost, the file system itself will always be consistent.

- **ZFS Pooled Storage**

  ZFS uses the concept of storage pools to manage physical storage. Historically, file systems were constructed on top of a single physical device. To address multiple devices and provide for data redundancy, the concept of a volume manager was introduced to provide the image of a single device so that file systems would not have to be modified to take advantage of multiple devices. This design added another layer of complexity. ZFS eliminates volume management altogether, aggregating devices into a storage pool. The storage pool describes the physical characteristics of the storage (device layout, data redundancy, and so on) and acts as an arbitrary data store from which file systems can be created. You no longer need to predetermine the size of a file system, as file systems grow automatically within the space allocated to the storage pool. When new storage is added, all file systems within the pool can immediately use the additional space without additional work.

- **Checksums and Self-Healing Data**

  With ZFS, all data and metadata is checked by checksum; ZFS also provides for self-healing data. ZFS supports storage pools with varying levels of data redundancy, including mirroring and a variation on RAID 5. When a bad data block is detected, ZFS fetches the correct data from another replicated copy, and repairs the bad data, replacing it with the good copy.

- **ZFS Snapshot™ Copies and Clones**

  A Snapshot copy is a read-only copy of a file system or volume. Initially, Snapshot copies consume no additional space within the pool. As data within the active dataset changes, the Snapshot copy consumes space by continuing to reference the old data.

  A clone is a file system whose initial contents are identical to the contents of a Snapshot copy.

- **Architectural Independence**

  A zpool with ZFS file systems and zvols created and used on a Sparc platform should be exportable to an x86 platform.

- **Simplified Administration**

  ZFS manages file systems through a hierarchy that allows for simplified management of properties such as quotas, reservations, compression, and mount points without needing multiple commands or editing configuration files. File systems themselves are very cheap (equivalent to a new directory), so you are encouraged to create a file system for each user, project, workspace, and so on.

2. Support for ZFS

ZFS was introduced in Solaris 10 Update 2. NetApp supports ZFS with Sun™ native drivers and multipathing with Data ONTAP® 7.2.2 and later running in single-image cfmode.

- **Multipathing**
  ZFS will require Sun native multipathing. Veritas™ DMP is not possible with ZFS because ZFS and Veritas are mutually exclusive per disk; a ZFS disk can't be a Veritas disk and therefore Veritas can't control the multiple I/O paths for a ZFS disk.
- **ZVOL**
  ZVOLs offer a way to provide raw disk space from a zpool. ZVOLS may be used to create legacy disk devices with legacy file systems or to provide raw space for applications.
- **Delegation of ZFS datasets to Zones**
  A ZFS dataset can be delegated to a Solaris zone. Then the zone administrator can manage the dataset and create child datasets as needed.

3. LUN type for ZFS

You can provision storage for ZFS with whole disks or slices. It is recommended that you do not have different file systems on different slices (for example, UFS on slice 0 and ZFS on slice 1) when you provision slices for ZFS. With NetApp storage array LUN's, provision an entire slice for use with ZFS, instead of the entire disk.

ZFS formats the disk using an EFI label to contain a single, large slice. ZFS applies an EFI label when you create a storage pool with whole disks. Disks can also be labeled with a traditional Solaris VTOC label when you create a storage pool with a disk slice.

There are two ways that you can provision a disk for use with ZFS.

- VTOC - Use the normal VTOC label and provision a single slice for use with ZFS. Create a slice that encompasses the entire disk and provision that single slice to ZFS.
- EFI – Invoke the advanced "format –e" command and label the disk with an EFI label. Create a slice starting from sector 40 so that it is aligned with WAFL® and provision that slice for use with ZFS.

There is no separate LUN type on the storage array for EFI labels or EFI offsets, so use the "Solaris" LUN type and manually align a slice encompassing the entire disk when using EFI labels. The LUN type "Solaris" is also used for the normal VTOC label.

If we provision the entire disk instead of a slice for use with ZFS, then ZFS will format the disk with an EFI label, which will be unaligned with WAFL. For this reason, it is recommended to use whole "aligned" slices (for EFI labels) that encompass the size of the disk for use with ZFS. We will have to manually align the slice and then provision the slice for use with ZFS if we want to use EFI labels. EFI disks add support for disks greater than 1TB. There is no need to "align" the slice if VTOC labels are used; just create a slice that is the entire size of the disk and provision that slice to ZFS.

An example of aligning a disk with an EFI label and then creating a zpool with the aligned slice is given below:

```
# format -e
Searching for disks...done
```

```
AVAILABLE DISK SELECTIONS:
       0. c1t0d0 <ATA-HITACHIHDS7280S-A6EA cyl 65533 alt 2 hd 16 sec 149>
          /pci@1e,600000/pci@0/pci@9/pci@0/scsi@1/sd@0,0
       1. c4t60A98000433461374E3442586F424849d0 <NETAPP-LUN-0.2-5.00GB>
          /scsi_vhci/ssd@g60a98000433461374e3442586f424849
       2. c4t60A98000433461374E3442586F436877d0 <NETAPP-LUN-0.2-8.01GB>
          /scsi_vhci/ssd@g60a98000433461374e3442586f436877
Specify disk (enter its number): 2
Selecting c4t60A98000433461374E3442586F436877d0
[disk formatted]

FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        inquiry    - show vendor, product, and revision
        scsi       - independent SCSI mode selects
        cache      - enable, disable, or query SCSI disk cache
        volname    - set 8-character volume name
        !<cmd>     - execute <cmd>, then return
        quit
format> p


PARTITION MENU:
        0      - change `0' partition
        1      - change `1' partition
        2      - change `2' partition
        3      - change `3' partition
        4      - change `4' partition
        5      - change `5' partition
        6      - change `6' partition
```

```
        7      - change `7' partition
        8      - change '8' partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        !<cmd> - execute <cmd>, then return
        quit
partition> p
Current partition table (original):
Total disk sectors available: 20964472 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm               34       10.00GB          20964472
  1 unassigned    wm                0           0               0
  2 unassigned    wm                0           0               0
  3 unassigned    wm                0           0               0
  4 unassigned    wm                0           0               0
  5 unassigned    wm                0           0               0
  6 unassigned    wm                0           0               0
  7 unassigned    wm                0           0               0
  8   reserved    wm         20964473        8.00MB          20980856

partition> m
Select partitioning base:
        0. Current partition table (original)
        1. All Free Hog
Choose base (enter number) [0]? 1

Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm                0           0               0
  1 unassigned    wm                0           0               0
  2 unassigned    wm                0           0               0
  3 unassigned    wm                0           0               0
  4 unassigned    wm                0           0               0
  5 unassigned    wm                0           0               0
  6 unassigned    wm                0           0               0
  7 unassigned    wm                0           0               0
  8   reserved    wm                0           0               0
```

```
Do you wish to continue creating a new partition
table based on above table[yes]?
Free Hog partition[6]?
Enter size of partition 0 [0b, 33e, 0mb, 0gb, 0tb]: 39e
Enter size of partition 1 [0b, 39e, 0mb, 0gb, 0tb]:
Enter size of partition 2 [0b, 39e, 0mb, 0gb, 0tb]:
Enter size of partition 3 [0b, 39e, 0mb, 0gb, 0tb]:
Enter size of partition 4 [0b, 39e, 0mb, 0gb, 0tb]:
Enter size of partition 5 [0b, 39e, 0mb, 0gb, 0tb]:
Enter size of partition 7 [0b, 39e, 0mb, 0gb, 0tb]:
Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm               34       0.00MB              39
  1 unassigned    wm                0            0               0
  2 unassigned    wm                0            0               0
  3 unassigned    wm                0            0               0
  4 unassigned    wm                0            0               0
  5 unassigned    wm                0            0               0
  6        usr    wm               40      10.00GB        20964471
  7 unassigned    wm                0            0               0
  8   reserved    wm         20964472       8.00MB        20980855

Ready to label disk, continue? y

partition>
```

Slice 6 is now "aligned" and ready for use with ZFS.

```
#zpool create ntap c4t60A98000433461374E3442586F436877d0s6
```

A simple stripe pool called "ntap" with slice 6 of size 10g is created.

To reiterate, a dummy slice including sector 39 is created (sector 0–39 is 40 sectors) and ZFS is given the "aligned" slice starting from sector 41. The "aligned" slice that starts from sector 41 onwards falls on a WAFL 4k boundary.

4. ZFS Storage Pools Recommendations

ZFS supports storage pools with varying levels of data redundancy, including mirroring and a variation on RAID 5. Zpools can be composed of a simple stripe, RAID Z, RAID Z2, or mirrored configurations. (RAID Z is a variant of RAID 5, and RAID Z2 is RAID Z with double parity.)

As an example, you could create the following configurations out of four disks:

_ Four disks using dynamic striping

_ One four-way RAID Z configuration

_ Two two-way mirrors using dynamic striping

One of the advantages of ZFS is its self-healing mechanisms. When a bad data block is detected, ZFS fetches the correct data from another replicated copy, and repairs the bad data, replacing it with the good copy. The self-healing mechanism works only when the zpools are redundant pools.

The ZFS administration guide section on data recovery also has a note of caution: if your pool is not replicated, the chance that data corruption can render some or all of your data inaccessible is always present. With this in mind, it is recommended to use redundant zpools to take advantage of the self-healing mechanism.

In line with best practices, in a replicated pool configuration, leverage multiple controllers to reduce hardware failures and improve performance. For example:

```
# zpool create ntap mirror c1t0d0s6 c2t0d0s6
```

Set up hot spares to speed up healing in the face of hardware failures. For example:

```
# zpool create ntap mirror c1t0d0s0 c2t0d0s0 spare c1t1d0s0 c2t1d0s0
```

You can also consider:

Multiple Storage Pools on the Same System

- You can pool resources into one ZFS storage pool, which allows different file systems to get the benefit from all the "pooled" resources. This can greatly increase the performance seen by any one file system. You can use multiple LUNs, provision them to the host, and create redundant zpools.
- If some workloads require more predictable performance characteristics, then the administrator could look at separating the loads into different pools.
- Currently, pool performance can degrade when a pool is very full and file systems are updated frequently, such as on a busy mail server. Under these circumstances, it is recommended to keep pool space below 80% utilization to maintain pool performance.

4.1 Should I Configure a RAID Z, RAID Z2, or a Mirrored Storage Pool?

As in any case, the general consideration is whether your goal is to maximize disk space or maximize performance.

- A RAID Z configuration maximizes disk space and generally performs well when data is written and read in large chunks (128K or more).
- A RAID Z2 configuration offers excellent data availability, and performs similarly to RAID Z. RAID Z2 has significantly better mean time to data loss (MTTDL) than either RAID Z or two-way mirrors.
- A mirrored configuration consumes more disk space but generally performs better with small random reads.
- If your I/Os are large, sequential, or write-mostly, then ZFS's I/O scheduler aggregates them in such a way that you'll get very efficient use of the disks regardless of the data replication model.
- For better performance, a mirrored configuration is strongly favored over a RAID Z configuration, particularly for large, uncacheable, random read loads.

For more information, see this blog:

http://blogs.sun.com/relling/entry/zfs_raid_recommendations_space_performance

5. LUN Expansion

Currently, ZFS cannot automatically handle LUN growth. If you have two LUNs in a mirror, and you grow both of them on the storage array, the mirror will not automatically grow to encompass the increased sizes of the LUNs. You will have to export the zpool, re-label the LUNs (using the new capacity), and import that zpool to be able to see the increased size. This also means downtime of the applications using the zpool, and is risk-prone, since it requires manual "re-labeling" of the LUNs. This method is not recommended for production machines.

A recommended method would be to add devices to the zpool instead of "resizing" the underlying devices. For example, if the zpool has a mirrored configuration, add two more LUNs to the configuration to "grow" the zpool and use the increased capacity. Think of this as an equivalent of adding drives to an aggregate.

Sun is currently testing a ZFS dynamic LUN expansion feature. This is expected to be available in the near future.

6. Clones

A clone is a writable volume or file system whose initial contents are the same as the dataset from which it was created. As with Snapshot copies, creating a clone is nearly instantaneous, and initially consumes no additional disk space. A clone does not inherit the properties of the dataset from which it was created. Rather, clones inherit their properties based on where the clones are created in the pool hierarchy.

To create a clone, use the ZFS clone command, specifying the Snapshot copy from which to create the clone, and the name of the new file system or volume. The new file system or volume can be located anywhere in the ZFS hierarchy and it cannot be in a pool that is different from where the original file system Snapshot copy resides. The type of the new dataset (for example, file system or volume) is the same type as the Snapshot copy from which the clone was created.

In the following example, a new clone named test/home/clone with the same initial contents as the Snapshot test/home/original@yesterday is created.

 # zfs snapshot test/share/original@yesterday

 # zfs clone test/share/original@yesterday test/home/clone

To replace the original ZFS file system with the clone, use the ZFS promote command. Check the ZFS administration guide for more details.

"Note: Clones can also be created on the NetApp storage array, but if the zpool has LUNs from multiple storage arrays and/or multiple volumes, the administrator must be cognizant to create clones of all NetApp volumes housing LUN's in the zpool. The administrator must also ensure that the clones are 'consistent,' which is discussed in the 'snapshot' section."

Important: The cloned LUN(s) on the array cannot be exposed to the same machine that has the original LUN(s).The host will complain about "duplicate" entities and will not import the zpool.

7. Snapshot Copies

A Snapshot copy is a read-only copy of a file system or volume. Snapshot copies can be created almost instantly, and initially consume no additional disk space within the pool. However, as data within the active dataset changes, the Snapshot copy consumes disk space by continuing to reference the old data and so prevents the space from being freed.

Snapshot copies are created by using the ZFS snapshot command, which takes as its only argument the name of the Snapshot copy to create. The Snapshot copy name is specified as follows:

filesystem@snapname

volume@snapname

In the following example, a Snapshot copy of fcp/netapp that is named monday is created.

 # zfs snapshot fcp/netapp/@monday

You can also create Snapshot copies for all descendant file systems by using the -r option. For example:

 # zfs snapshot -r fcp/netapp@now

The ZFS snapshot creation process flushes the file system before taking the snapshot. For example, With databases running on ZFS, the procedure for db and FS consistency would be:

1 Quiesce the application.

2 Put the db in hot backup mode – This ensures db consistency by not allowing any further writes during this activity.

3 Type in "sync" at the prompt to flush all outstanding writes.

4 Take a ZFS Snapshot copy – This also flushes any outstanding writes to ensure file system consistency.

5 Take the db out of hot backup mode.

Step 4 - Take a ZFS Snapshot copy – This activity also does an implicit "sync" in the background, which is mentioned explicitly in step 3, but this is just to reiterate the fact that all writes have to be flushed from the file system for it to be consistent.

Note: Continuing from the "Clone" section above, it is recommended to take a ZFS Snapshot copy before taking a Snapshot copy on the array so that we have a consistent "LUN" for cloning purposes. Also, keep in mind that ZFS file system Snapshot copies would not be useful since it would flush only specific file systems, whereas LUN's might have multiple file systems spanning them. This might lead to inconsistent ZFS file systems if a SnapRestore® process is carried out, so it is recommended to take ZFS volume level Snapshot copies where possible to ensure "LUN" level consistency. For example, if a zpool has five LUNs out of which three are in one FlexVol® volume and the remaining two are in another FlexVol volume, then you will have to take a ZFS volume level Snapshot copy to flush out data to all five devices in the zpool. You would then take Snapshot copies of the two FlexVol volumes that contain the five LUNs and restore them with SnapRestore when the need arises. This also means that all of the ZFS file systems will be reverted to that point in time when the array Snapshot copy was taken. For more granular control of the ZFS file system, use ZFS Snapshot copies.

8. References

1. The ZFS administration guide:

http://docs.sun.com/app/docs/doc/819-5461

2. Some excellent blogs about various topics in ZFS:

http://www.opensolaris.org/os/community/zfs/blogs/;jsessionid=2F77646DBE117B732105701044D24D97

3. ZFS and containers:

http://www.sun.com/software/solaris/howtoguides/zfshowto.jsp

4. Hosting databases on ZFS:

http://blogs.sun.com/realneel/entry/zfs_and_databases

5. ZFS raid recommendations – space or performance:

http://blogs.sun.com/relling/entry/zfs_raid_recommendations_space_performance

6. ZFS best practices from Solaris internals:

http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

9. Acknowledgements

The author would like to thank the following individuals at Network Appliance, Inc. for their contribution to this technical report:

Victor Engle, Peter Kearney

10. Disclaimer

Each environment has its own specific set of requirements and no guarantees can be given that the results presented in this report will work as expected on other platforms. This paper should assist in the research and troubleshooting that may be required in a particular case and serve as a checklist of items to be aware of. Please forward any errors, omissions, differences, new discoveries, and comments about this paper to ganesh.kamath@netapp.com.